Analysis of existing design

Problem 1:The current SpaceInvader class uses “if else” statements to control different behaviors of different Alien types, since alienType is stored as a String in class Alien. This is a high coupling between SpaceInvader and Alien, and low cohesion within the class Alien. This would also result in poor extensibility as adding/removing alien types would require modification in both Alien and SpaceInvader classes.
Solution: Use polymorphism where each Alien type is a subclass to an abstract parent class Alien so that each child Alien class deals with their own logic independently. Consequently, SpaceInvader would have less coupling with Alien, and also the design would be more open for Alien extensions where a new alien type can simply become a new child class of Alien, and new features for each type of alien can be done within their own child class without interfering with the logic of other types of aliens.

Problem 2: The SpaceInvader class has low cohesion as it is bloated with responsibilities:

- Run app
- Create spaceship and aliens
    - By Information Expert, this responsibility is suitable.
- Act
- Know the input - key press
    - A Controller can be created for this responsibility instead so that it is the first object behind the UI layer.
- Listener to alien hit
    - To increase cohesion, this responsibility could be assigned to a purely fabricated listener class.

Solution: Delegate creation of aliens to a fabricated class; delegate bomb hit alien logic to a separate fabricated class to reduce the number of responsibilities in the SpaceInvader class.

Proposed new design of the simple version

- A fabricated class to handle alien creation - moves responsibility from SpaceInvader to AlienController, so that SpaceInvader has higher cohesion and lower coupling.
- A fabricated class that handles alien hit logics - move this responsibility from Bomb to fabricated AlienHit for higher cohesion and lower coupling between Bomb and Alien.
- Incorporate inheritance polymorphism by creating the Alien class as an abstract parent class, with NormalAlien and potentially plus version aliens extending from Alien. This polymorphism would allow better alien extensions, because each additional type of alien can be a subclass of Alien sharing some common functionalities and also having different independent logics specific to their alien type utilizing subclasses.

Proposed design of the extended version

- Implementation of a package for Alien class types to enhance maintainability as well as data encapsulation to provide controlled access.
- We fabricated an AlienController class that is responsible for Alien creation by Controller and Pure Fabrication pattern, that creates the new aliens through the method setUpAliens(...), and channels timely information to MultipleAlien. The AlienController class also acts as a medium of indirection between SpaceInvader gameGrid and MultipleAlien for exchange of data for MultipleAlien multiplication.
- We also fabricated a HitAlien class that deals with the collision logic between bomb and alien, responsible for checking the status of the alien shot, whether the bomb hit is on visible or invulnerable alien, and reducing alien life if its status is not invulnerable and is visible (through calling alien.decrementLife()), checking if the alien is killed on shot by evaluating alien.getNbLives()==0, and creating an explosion when an alien is killed (through calling explode()), etc.
- We changed Alien class into an abstract class with concrete subclasses NormalAlien, PowerfulAlien, Invulnerable Alien and Multiple Alien, by the principle of polymorphism. PowerfulAlien’s nbLives is initialised to be 5 to accommodate its powerful feature, resistable against 4 extra bullet shots. InvulnerableAlien randomly switches to being Invulnerable in its act() method overrode and has an invulnerabilityTimer in its own subclass that controls the period of invulnerability. MultipleAlien has an extra multiply() method which uses the turnIntoNormal() method and turns the lowest top row of invisible normal aliens into visible, with timely alienGrid channeled indirectly from SpaceInvader through middle class AlienController, to achieve lower coupling with SpaceInvader.
- We delegated the SpaceShip class the responsibility of modifying alien speed based on the number of shots fired, through the switch statements in the act() method, which branches by the number of shots fired and calls Alien.setStepSize(int n) accordingly. This delegation of “doing” responsibility is made by Information Expert, as SpaceShip stores the information we need, the number of shots fired.
  (Alternatively, we could have fabricated an AlienSpeedControl class that is responsible for listening to / knowing the number of shots fired and adjusting the alien speeds accordingly, which would allow more flexibility in extensions to alien speed. However, we judged that would be unnecessary coupling since it is unlikely to have another variable affecting alien speed besides number of shots fired. In this case, it would be neater to delegate alien speed control directly to SpaceShip.)

We created a domain class diagram (Figure 1), a design class diagram (Figure 2), partial design sequence diagrams (Figure 3.1, 3.2) and state machine diagrams (Figure 4.1, 4.2, 4.3) to further illustrate the design of this extended version, as shown in the following pages.

Models for the extended version

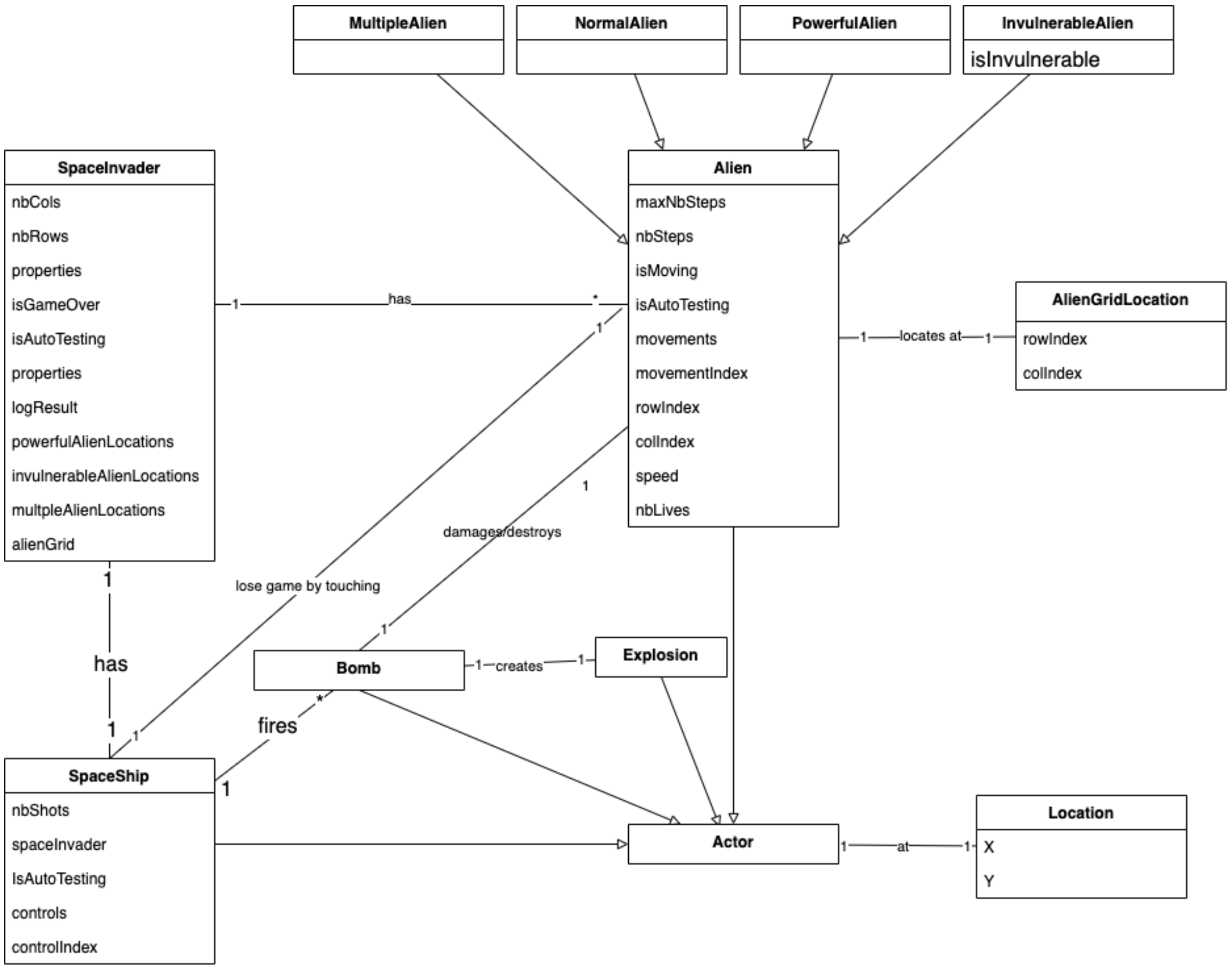

Figure 1: A domain class diagram showing the noteworthy concepts and their association in Space Invaders Plus including the extensions. Key extra concepts: “speed” and “nbLives” in Alien, as well as multiple subclasses extending from Alien.

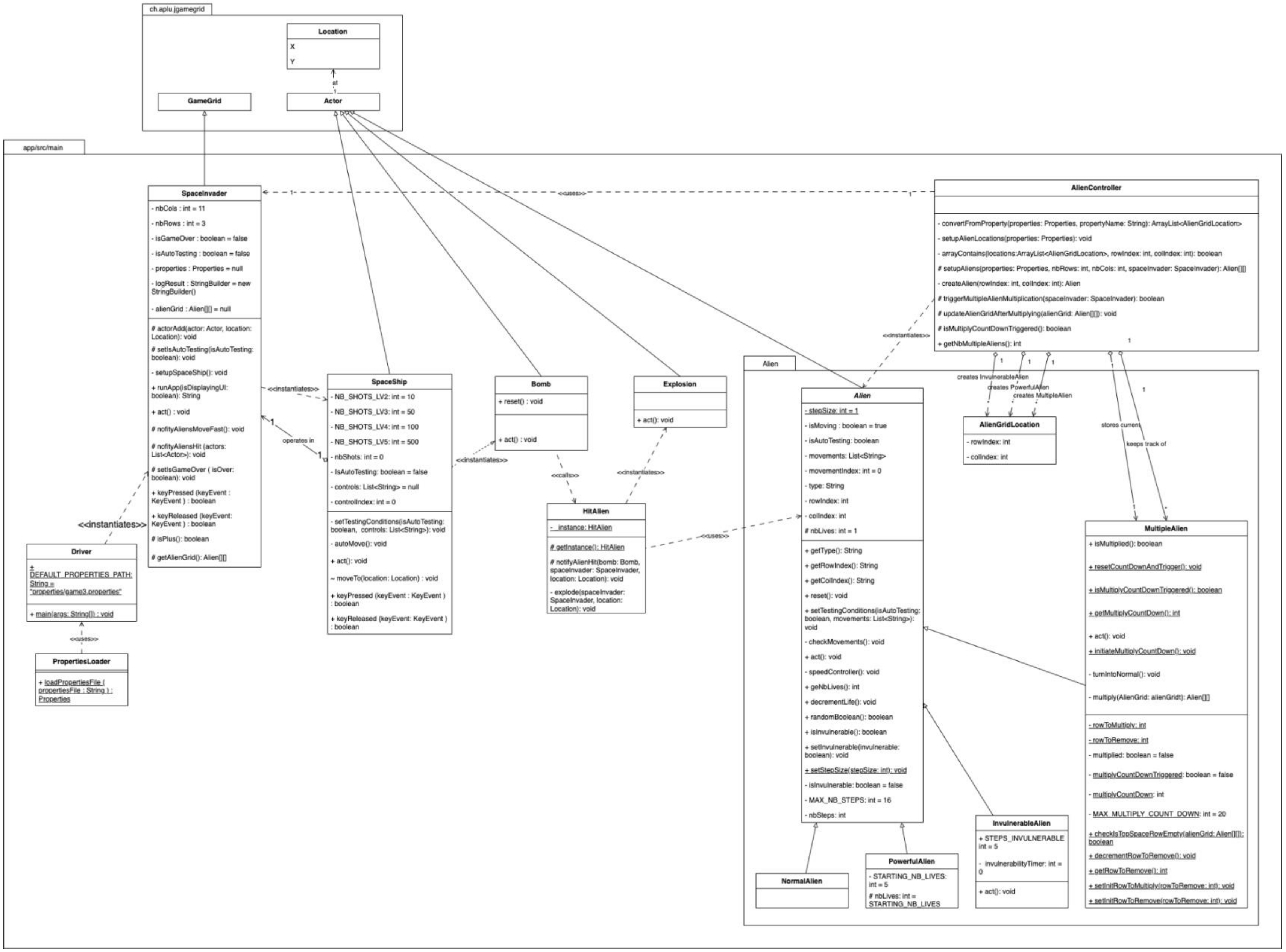
ch.aplu.jgamegrid
Location
X
Y
at
GameGrid
Actor
app/src/main
SpaceInvader
- nbCols : int = 11
- nbRows : int = 3
- isGameOver : boolean = false
- isAutoTesting : boolean = false
- properties : Properties = null
- logResult : StringBuilder = new StringBuilder()
- alienGrid : Alien[][] = null
# actorAdd(actor: Actor, location: Location): void
# setIsAutoTesting(isAutoTesting: boolean): void
- setupSpaceShip(): void
+ runApp(isDisplayingUI: boolean): String
+ act() : void
# nofityAliensMoveFast(): void
# nofityAliensHit (actors: List<Actor>): void
# setIsGameOver (isOver: boolean): void
+ keyPressed (keyEvent : KeyEvent) : boolean
+ keyReleased (keyEvent: KeyEvent) : boolean
# isPlus(): boolean
# getAlienGrid(): Alien[][]
<<uses>>
1
1
<<instantiates>>
operates in
1
1
<<instantiates>>
Driver
+ DEFAULT_PROPERTIES_PATH: String = "properties/game3.properties"
+ main(args: String[]) : void
<<uses>>
PropertiesLoader
+ loadPropertiesFile (propertiesFile : String) : Properties
SpaceShip
- NB_SHOTS_LV2: int = 10
- NB_SHOTS_LV3: int = 50
- NB_SHOTS_LV4: int = 100
- NB_SHOTS_LV5: int = 500
- nbShots: int = 0
- IsAutoTesting: boolean = false
- controls: List<String> = null
- controlIndex: int = 0
- setTestingConditions(isAutoTesting: boolean, controls: List<String>): void
- autoMove(): void
+ act(): void
~ moveTo(location: Location) : void
+ keyPressed (keyEvent : KeyEvent) : boolean
+ keyReleased (keyEvent: KeyEvent) : boolean
<<instantiates>>
Bomb
+ reset() : void
+ act() : void
<<calls>>
Explosion
+ act(): void
<<instantiates>>
HitAlien
- _instance: HitAlien
# getInstance(): HitAlien
# notifyAlienHit(bomb: Bomb, spaceInvader: SpaceInvader, location: Location): void
- explode(spaceInvader: SpaceInvader, location: Location): void
<<uses>>
AlienController
- convertFromProperty(properties: Properties, propertyName: String): ArrayList<AlienGridLocation>
- setupAlienLocations(properties: Properties): void
- arrayContains(locations:ArrayList<AlienGridLocation>, rowIndex: int, colIndex: int): boolean
# setupAliens(properties: Properties, nbRows: int, nbCols: int, spaceInvader: SpaceInvader): Alien[][]
- createAlien(rowIndex: int, colIndex: int): Alien
# triggerMultipleAlienMultiplication(spaceInvader: SpaceInvader): boolean
# updateAlienGridAfterMultiplying(alienGrid: Alien[][]): void
# isMultiplyCountDownTriggered(): boolean
+ getNbMultipleAliens(): int
<<instantiates>>
1
1
1
1
1
1
creates InvulnerableAlien
creates PowerfulAlien
creates MultipleAlien
AlienGridLocation
- rowIndex: int
- colIndex: int
stores current
keeps track of
Alien
Alien
- stepSize: int = 1
- isMoving : boolean = true
- isAutoTesting: boolean
- movements: List<String>
- movementIndex: int = 0
- type: String
- rowIndex: int
- colIndex: int
# nbLives: int = 1
+ getType(): String
+ getRowIndex(): String
+ getColIndex(): String
+ reset(): void
+ setTestingConditions(isAutoTesting: boolean, movements: List<String>): void
- checkMovements(): void
+ act(): void
- speedController(): void
+ geNbLives(): int
+ decrementLife(): void
+ randomBoolean(): boolean
+ isInvulnerable(): boolean
+ setInvulnerable(invulnerable: boolean): void
+ setStepSize(stepSize: int): void
- isInvulnerable: boolean = false
- MAX_NB_STEPS: int = 16
- nbSteps: int
NormalAlien
PowerfulAlien
- STARTING_NB_LIVES: int = 5
# nbLives: int = STARTING_NB_LIVES
InvulnerableAlien
+ STEPS_INVULNERABLE int = 5
- invulnerabilityTimer: int = 0
+ act(): void
MultipleAlien
+ isMultiplied(): boolean
+ resetCountDownAndTrigger(): void
+ isMultiplyCountDownTriggered(): boolean
+ getMultiplyCountDown(): int
+ act(): void
+ initiateMultiplyCountDown(): void
- turnIntoNormal(): void
- multiply(AlienGrid: alienGridt): Alien[][]
- rowToMultiply: int
- rowToRemove: int
- multiplied: boolean = false
- multiplyCountDownTriggered: boolean = false
- multiplyCountDown: int
- MAX_MULTIPLY_COUNT_DOWN: int = 20
+ checkIsTopSpaceRowEmpty(alienGrid: Alien[][]): boolean
+ decrementRowToRemove(): void
+ getRowToRemove(): int
+ setInitRowToMultiply(rowToRemove: int): void
+ setInitRowToRemove(rowToRemove: int): void

Figure 2: A design class diagram for documenting the new design for Space Invaders Plus including the extensions. Some key implementations: fabricated HitAlien class to reduce coupling between Bomb, Explosion, Alien based. Indirection between Spaceship and Alien's hit logic through Bomb and HitAlien, as well as indirection between their creation logic through AlienController.

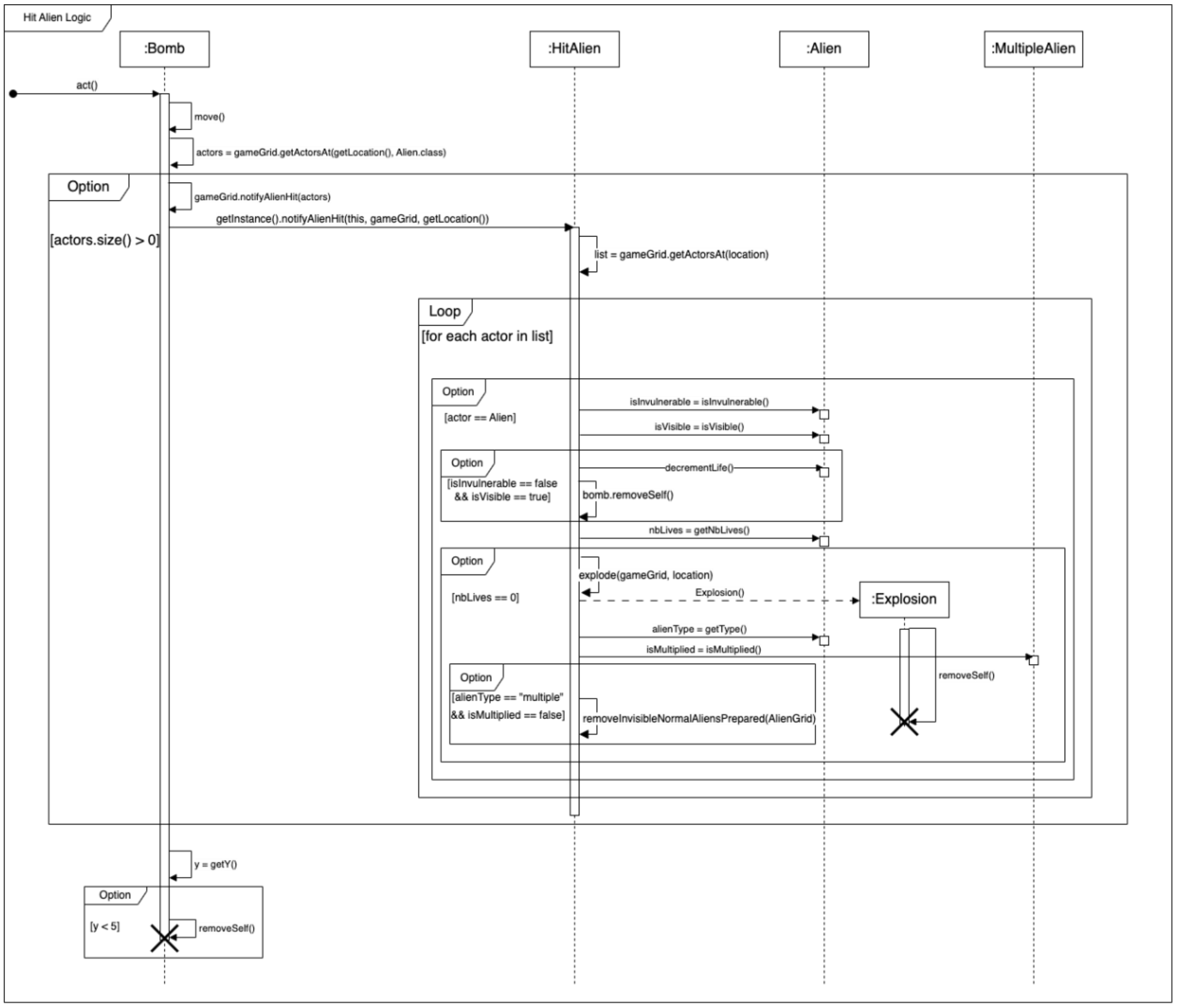
Hit Alien Logic
:Bomb
:HitAlien
:Alien
:MultipleAlien
act()
move()
actors = gameGrid.getActorsAt(getLocation(), Alien.class)
Option
[actors.size() > 0]
gameGrid.notifyAlienHit(actors)
getInstance().notifyAlienHit(this, gameGrid, getLocation())
list = gameGrid.getActorsAt(location)
Loop
[for each actor in list]
Option
[actor == Alien]
isInvulnerable = isInvulnerable()
isVisible = isVisible()
Option
[isInvulnerable == false && isVisible == true]
decrementLife()
bomb.removeSelf()
nbLives = getNbLives()
Option
[nbLives == 0]
explode(gameGrid, location)
Explosion()
:Explosion
alienType = getType()
isMultiplied = isMultiplied()
removeSelf()
Option
[alienType == "multiple" && isMultiplied == false]
removeInvisibleNormalAliensPrepared(AlienGrid)
y = getY()
Option
[y < 5]
removeSelf()

Figure 3.1: A partial design sequence diagram for documenting the dynamic behavior of aliens in terms of their being shot logic with respect to their alien type.

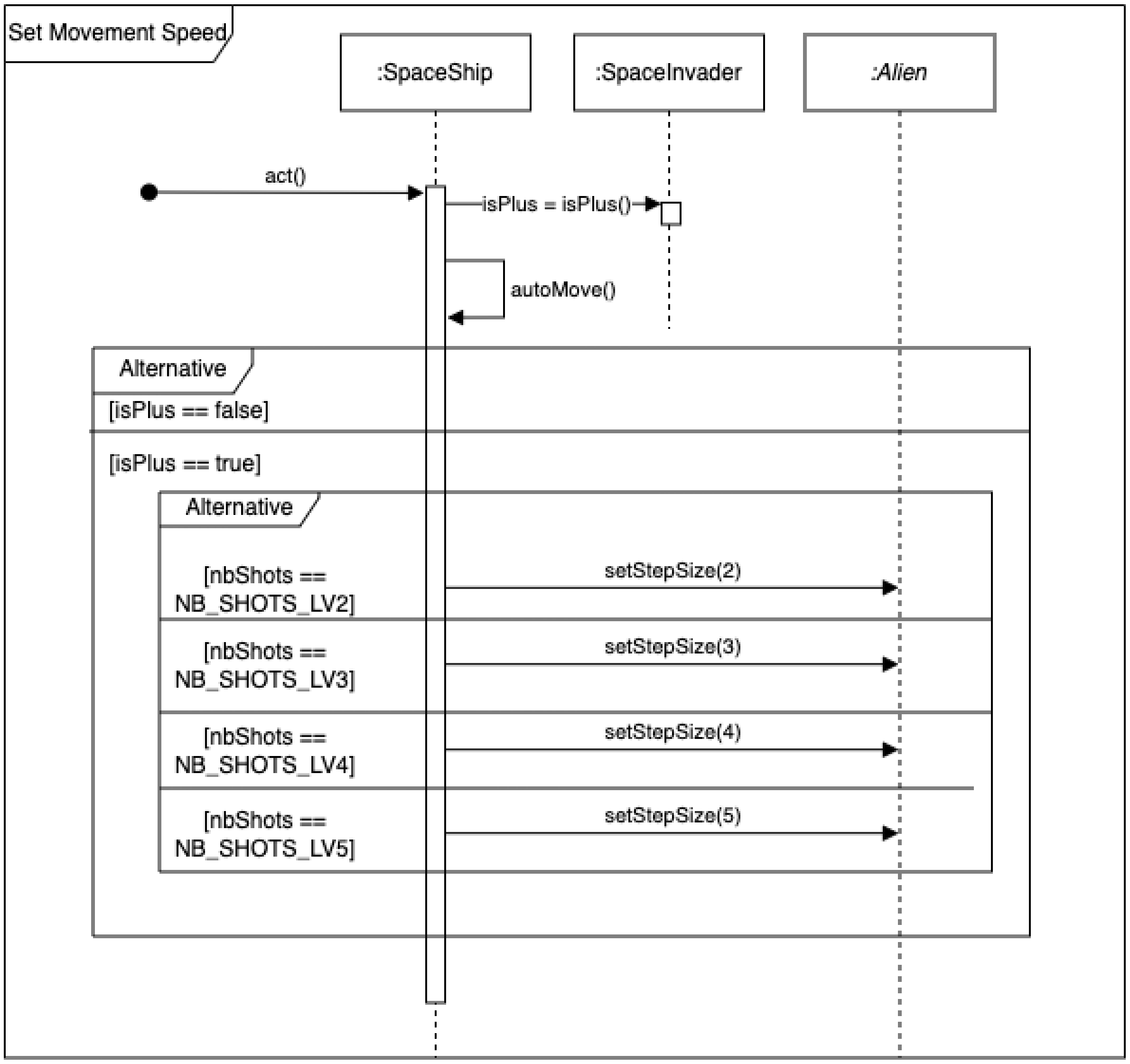

Figure 3.2: A partial design sequence diagram for documenting the dynamic behavior of aliens in terms of their speed in response to the number of shots as described in Feature 2 of Project Spec Section 1.2.

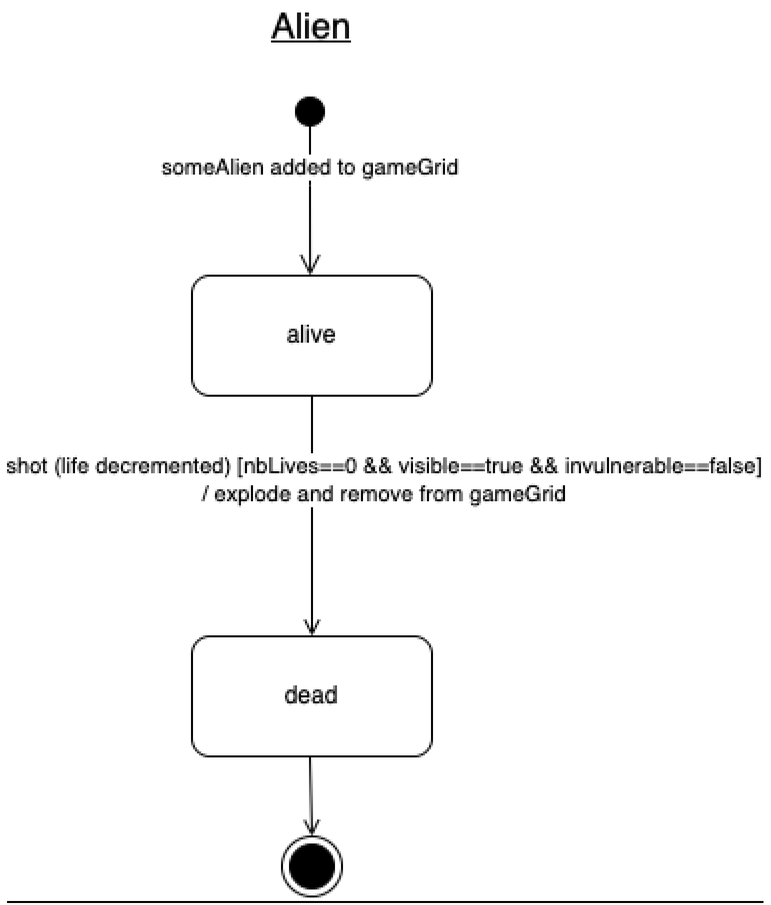
Alien
someAlien added to gameGrid
alive
shot (life decremented) [nbLives==0 && visible==true && invulnerable==false]
/ explode and remove from gameGrid
dead

Figure 4.1: State machine diagram showing how an alien's status transitions from alive to dead in general.

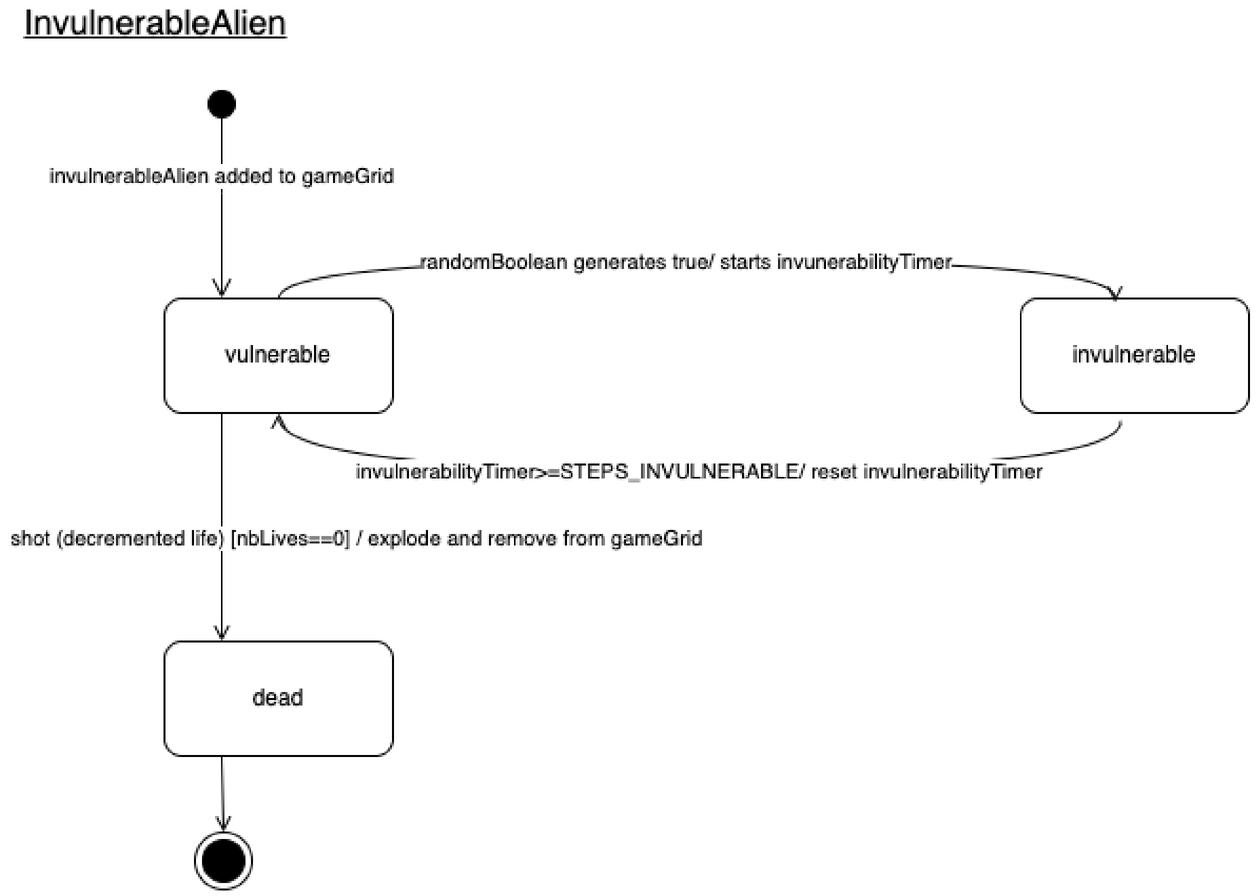
InvulnerableAlien
invulnerableAlien added to gameGrid
randomBoolean generates true/ starts invunerabilityTimer
vulnerable
invulnerable
invulnerabilityTimer>=STEPS_INVULNERABLE/ reset invulnerabilityTimer
shot (decremented life) [nbLives==0] / explode and remove from gameGrid
dead

Figure 4.2: State machine diagram showing the statuses of an InvulnerableAlien.

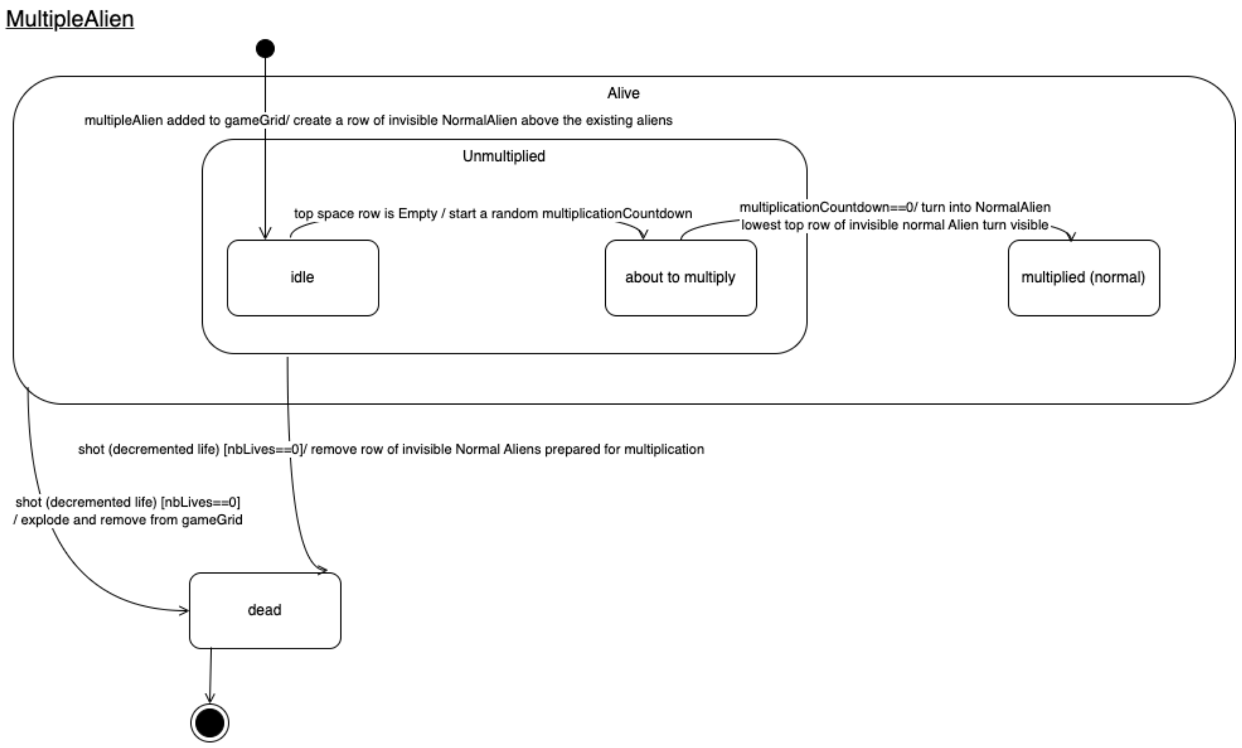
MultipleAlien
Alive
multipleAlien added to gameGrid/ create a row of invisible NormalAlien above the existing aliens
Unmultiplied
top space row is Empty / start a random multiplicationCountdown
multiplicationCountdown==0/ turn into NormalAlien
lowest top row of invisible normal Alien turn visible
idle
about to multiply
multiplied (normal)
shot (decremented life) [nbLives==0]/ remove row of invisible Normal Aliens prepared for multiplication
shot (decremented life) [nbLives==0]
/ explode and remove from gameGrid
dead

Figure 4.3: State machine diagram showing the statuses of a MultipleAlien.